

Contents

1 Routine/Function Prologues	2
1.0.1 iniTimeConst.F90 (Source File: iniTimeConst.F90)	2

1 Routine/Function Prologues

1.0.1 iniTimeConst.F90 (Source File: iniTimeConst.F90)

Initialize time invariant clm variables

Method:

Author: Gordon Bonan

USES:

```
use precision
use infnan
use clm_varder
use clm_varpar , only : nlevsoi, nlevlak
use clm_varmap , only : begpatch, endpatch, patchvec, numpatch
use clm_varcon , only : istsoil, istice, istdlak, istslak, istwet, spval
use pft_varcon , only : ncorn, nwheat, roota_par, rootb_par, &
                      z0mr, displar, dleaf, rhol, rhos, taul, taus, xl, &
                      qe25, vcmx25, mp, c3psn
use clm_varsur , only : zlak, dzlak, zsoi, dzsoi, zisoi

use clm_varctl , only : nsrest
use time_manager , only : get_step_size
use shr_const_mod, only : SHR_CONST_PI
use spmdMod      , only : masterproc

use grid_module      ! LIS Grid variables
use lisdrv_module
use lis_openfileMod
use lis_indices_module
```

CONTENTS:

```
! -----
! Initialize local variables for error checking
! -----


sand(:,:) = inf
clay(:,:) = inf

! -----
! itypveg, isoicol, itypwat, sand, and clay from 2-d surface type
! LIS modification: Read in soil files and equate lat,lon
! LIS grid arrays with clm grid
! Reynolds soil parameterization scheme
! Read in soil data maps to gridded arrays
! -----


print*, 'MSG: iniTimeConst -- Reading soil, clay and color files'
print*, 'MSG: iniTimeConst -- Opening soil file ',lis%p%safile
```

```

call lis_open_file(11, file=lis%p%safile, form='unformatted', &
                  status='old', script='getsand.pl')
print*, 'MSG: iniTimeConst -- Opening clay file ',lis%p%clfle
call lis_open_file(12, file=lis%p%clfle, form='unformatted', &
                  status='old', script='getclay.pl')
print*, 'MSG: iniTimeConst -- Opening color file ',lis%p%iscfile
call lis_open_file(13, file=lis%p%iscfile, form='unformatted', &
                  status='old', script='getcolor.pl')
read(11) sand1
read(12) clay1
read(13) color
close(11)
close(12)
close(13)
print*, 'MSG: iniTimeConst -- Read soil, clay and color files'
do k = 1,numpatch
  if (tile(k)%fgrd > 0.0) then      !valid subgrid patch
    clm(k)%kpatch = k
    clm(k)%itypveg = tile(k)%vegt
    !clm(k)%isoicol = color(tile(k)%col,tile(k)%row)
    clm(k)%isoicol = int(color(tile(k)%col,tile(k)%row-lis_tnroffset))
    if (tile(k)%vegt == npatch_urban) then !urban, from pcturb
      clm(k)%itypwat = istsoil
      print*, 'ERR: iniTimeConst -- not supposed to be in urban block'
    else if (tile(k)%vegt == npatch_lake) then !deep lake, from pctlak
      clm(k)%itypwat = istdlak
      do l = 1,nlevsoi
        sand(l,k) = 0._r8
        clay(l,k) = 0._r8
      end do
    else if (tile(k)%vegt == npatch_wet) then   !wetland, from pctwet
      clm(k)%itypwat = istwet
      do l = 1,nlevsoi
        sand(l,k) = 0._r8
        clay(l,k) = 0._r8
      end do
    else if (tile(k)%vegt == npatch_gla) then   !glacier, from pctgla
      clm(k)%itypwat = istice
      print*, 'ERR: iniTimeConst -- not supposed to be in glacier block'
    else                                     !soil
      clm(k)%itypwat = istsoil
      do l = 1, nlevsoi
        sand(l,k)=sand1(tile(k)%col,tile(k)%row-lis_tnroffset)
        clay(l,k)=clay1(tile(k)%col,tile(k)%row-lis_tnroffset)
      enddo
    end if
  end if
enddo

```

```
! -----
! tag lake points
! -----
```

```
do k = 1,numpatch
  if (clm(k)%itypwat==istdlak .or. clm(k)%itypwat==istslak) then
    clm(k)%lakpoi = .true.
  else
    clm(k)%lakpoi = .false.
  end if
end do
```

```
! -----
! latitudes and longitudes
! -----
```

```
pi = SHR_CONST_PI
do i=1,lis_nc_working
  do j=1,lis_nr_working
    if(gindex(i,j).ne. -1) then
      k = gindex(i,j)
      clm(k)%lat    = grid(tile(k)%index-lis_grid_offset)%lat * pi/180.
      clm(k)%lon    = grid(tile(k)%index-lis_grid_offset)%lon * pi/180.
    endif
  end do
enddo
```

```
! -----
! Define layer structure for soil and lakes
! Vertical profile of snow is initialized in routine iniTimeVar
! -----
```

```
if (nlevlak /= nlevsoi) then
  write(6,*)'number of soil levels and number of lake levels must be the same'
  write(6,*)'nlevsoi= ',nlevsoi,' nlevlak= ',nlevlak
  call endrun
endif
```

```
dzlak(1) = 1.
dzlak(2) = 2.
dzlak(3) = 3.
dzlak(4) = 4.
dzlak(5) = 5.
dzlak(6) = 7.
dzlak(7) = 7.
dzlak(8) = 7.
dzlak(9) = 7.
dzlak(10)= 7.
```

```

zlak(1) = 0.5
zlak(2) = 1.5
zlak(3) = 4.5
zlak(4) = 8.0
zlak(5) = 12.5
zlak(6) = 18.5
zlak(7) = 25.5
zlak(8) = 32.5
zlak(9) = 39.5
zlak(10)= 46.5

do j = 1, nlevsoi
    zsoi(j) = scalez*(exp(0.5*(j-0.5))-1.)      !node depths
enddo

dzsoi(1) = 0.5*(zsoi(1)+zsoi(2))              !thickness b/n two interfaces
do j = 2,nlevsoi-1
    dzsoi(j)= 0.5*(zsoi(j+1)-zsoi(j-1))
enddo
dzsoi(nlevsoi) = zsoi(nlevsoi)-zsoi(nlevsoi-1)

zisoi(0) = 0.
do j = 1, nlevsoi-1
    zisoi(j) = 0.5*(zsoi(j)+zsoi(j+1))          !interface depths
enddo
zisoi(nlevsoi) = zsoi(nlevsoi) + 0.5*dzsoi(nlevsoi)

do k = 1,numpatch
    if (clm(k)%itypwat == istdlak) then           !assume all lakes are deep lakes
        clm(k)%z(1:nlevlak) = zlak(1:nlevlak)
        clm(k)%dz(1:nlevlak) = dzlak(1:nlevlak)
    else if (clm(k)%itypwat == istslak) then      !shallow lake (not used)
        clm(k)%dz(1:nlevlak) = NaN
        clm(k)%z(1:nlevlak) = NaN
    else                                         !soil, ice, wetland
        clm(k)%z(1:nlevsoi) = zsoi(1:nlevsoi)
        clm(k)%dz(1:nlevsoi) = dzsoi(1:nlevsoi)
        clm(k)%zi(0:nlevsoi) = zisoi(0:nlevsoi)
    endif
end do

! -----
! Initialize root fraction (computing from surface, d is depth in meter):
! Y = 1 -1/2 (exp(-ad)+exp(-bd)) under the constraint that
! Y(d =0.1m) = 1-beta^(10 cm) and Y(d=d_obs)=0.99 with beta & d_obs
! given in Zeng et al. (1998).
! -----

```

```

! do k = begpatch, endpatch
do k = 1,numpatch
  if (.not. clm(k)%lakpoi) then
    ivt = clm(k)%itypveg
    do j = 1, nlevsoi-1
      clm(k)%rootfr(j) = .5*( exp(-roota_par(ivt)*clm(k)%zi(j-1)) &
                                + exp(-rootb_par(ivt)*clm(k)%zi(j-1)) &
                                - exp(-roota_par(ivt)*clm(k)%zi(j )) &
                                - exp(-rootb_par(ivt)*clm(k)%zi(j )) )
    end do
    clm(k)%rootfr(nlevsoi) = .5*( exp(-roota_par(ivt)*clm(k)%zi(nlevsoi-1)) &
                                + exp(-rootb_par(ivt)*clm(k)%zi(nlevsoi-1)) )

  else
    clm(k)%rootfr(1:nlevsoi) = spval
  end if
end do

! -----
! Initialize soil thermal and hydraulic properties
! -----
do k = 1,numpatch
  if (clm(k)%itypwat == istsoil) then           !soil
    do j = 1, nlevsoi
      clm(k)%bsw(j)     = 2.91 + 0.159*clay(j,k)*100.0
      clm(k)%watsat(j) = 0.489 - 0.00126*sand(j,k)*100.0
      xksat             = 0.0070556 *( 10.**(-0.884+0.0153*sand(j,k)*100.0) ) ! mm/s
      clm(k)%hksat(j)  = xksat * exp(-clm(k)%zi(j)/hkdepth)
      clm(k)%sucsat(j) = 10. * ( 10.**((1.88-0.0131*sand(j,k)*100.0) )
      tkm                = (8.80*sand(j,k)*100.0+2.92*clay(j,k)*100.0)/(sand(j,k)*100.0+clay(j,k)*100.0)
      bd                 = (1.-clm(k)%watsat(j))*2.7e3
      clm(k)%tkmg(j)   = tkm ** (1.- clm(k)%watsat(j))
      clm(k)%tksatu(j) = clm(k)%tkmg(j)*0.57**clm(k)%watsat(j)
      clm(k)%tkdry(j)  = (0.135*bd + 64.7) / (2.7e3 - 0.947*bd)
      clm(k)%csol(j)   = (2.128*sand(j,k)*100.0+2.385*clay(j,k)*100.0)/ (sand(j,k)*100.0+clay(j,k)*100.0)
    end do
  else                                         !ice, lakes, wetlands
    do j = 1, nlevsoi
      clm(k)%bsw(j)     = spval
      clm(k)%watsat(j) = spval
      clm(k)%hksat(j)  = spval
      clm(k)%sucsat(j) = spval
      clm(k)%tkmg(j)   = spval
      clm(k)%tksatu(j) = spval
      clm(k)%tkdry(j)  = spval
      clm(k)%csol(j)   = spval
    end do
  end if

```

```

end do

! -----
! Initialize clm derived type components from pft_varcon to avoid
! indirect addressing and be compatible with offline CLM code
! -----

! do k = begpatch, endpatch
do k = 1,numpatch
  ivt = clm(k)%itypveg
  clm(k)%z0mr      = z0mr(ivt)
  clm(k)%displar = displar(ivt)
  clm(k)%dleaf   = dleaf(ivt)
  clm(k)%xl       = xl(ivt)
  do ib = 1,numrad
    clm(k)%rhol(ib) = rhol(ivt,ib)
    clm(k)%rhos(ib) = rhos(ivt,ib)
    clm(k)%taul(ib) = taul(ivt,ib)
    clm(k)%taus(ib) = taus(ivt,ib)
  end do
  clm(k)%qe25      = qe25(ivt)          ! quantum efficiency at 25c (umol co2 / umol photon)
  clm(k)%vcmx25   = vcmx25(ivt)        ! maximum rate of carboxylation at 25c (umol co2/m**2/s)
  clm(k)%mp        = mp(ivt)            ! slope for conductance-to-photosynthesis relationship
  clm(k)%c3psn    = c3psn(ivt)          ! photosynthetic pathway: 0. = c4, 1. = c3
end do

!Initialize other misc derived type components - note that for a
!restart run, dtime is set in routine restrd()

if (nsrest == 0) then
!   do k = begpatch, endpatch
  if ( masterproc ) then
    tmp_dtime = get_step_size()
  endif
#if ( defined OPENDAP )
  call MPI_BCAST(tmp_dtime,1,MPI_REAL,0,MPI_COMM_WORLD,ierr)
#endif
  do k = 1,numpatch
    !clm(k)%dtime = get_step_size()
    clm(k)%dtime = tmp_dtime
  end do
endif

print*, "DBG: iniTimeConst -- dtime", clm(1)%dtime, (',iam,')

if (masterproc) then
  write(6,*)
  write(6,30)

```

```
do j = 1,nlevlak
    write(6,40)zlak(j),dzlak(j)
end do
write(6,*)
write(6,35)
do j = 1,nlevsoi
    write(6,45)zsoi(j),dzsoi(j),zisoi(j)
end do
write(6,50)
write(6,*)
endif

30 format(' ',' lake levels ',' lake thickness(m)')
35 format(' ',' soil levels ',' soil thickness(m)', ' soil interfaces(m)')
40 format(' ',2(f7.3,8x))
45 format(' ',3(f7.3,8x))
50 format(' ','Note: top level soil interface is set to 0')

return
```